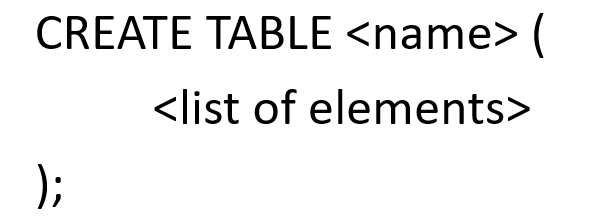
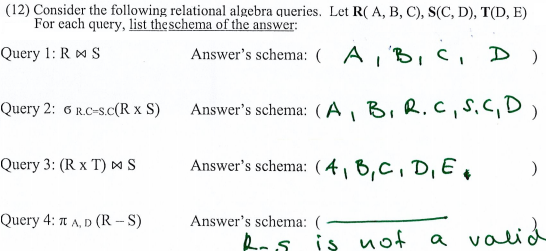
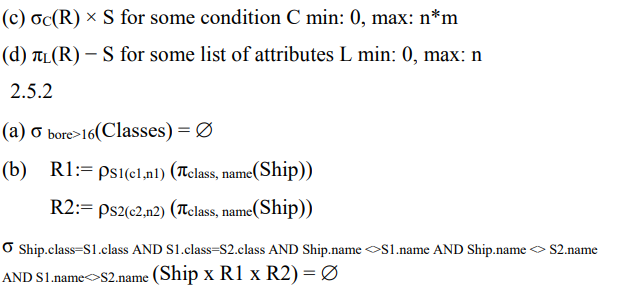
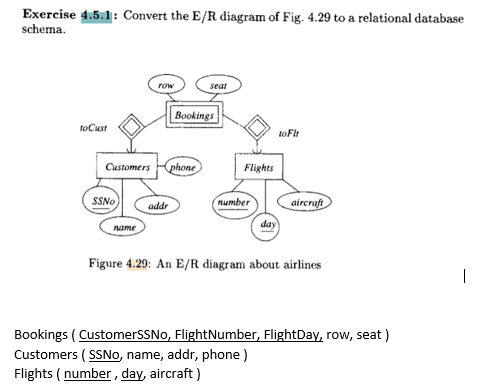
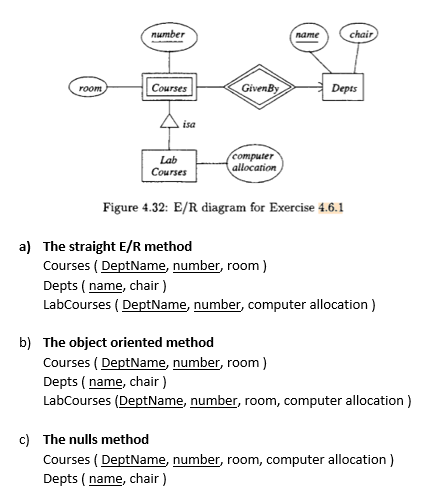
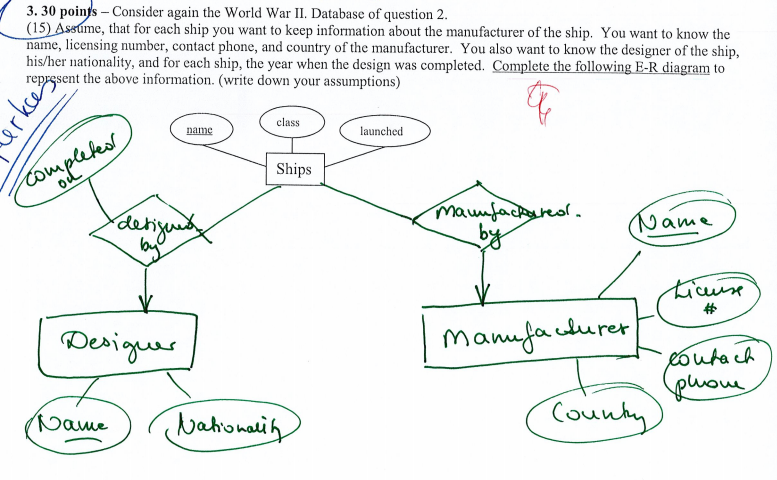
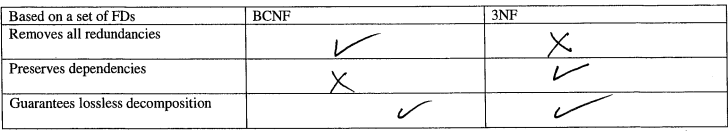
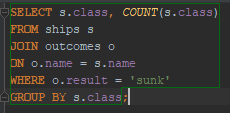
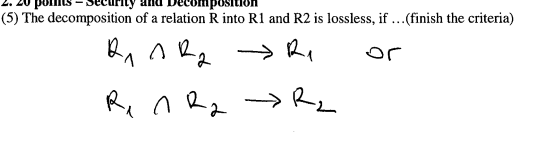
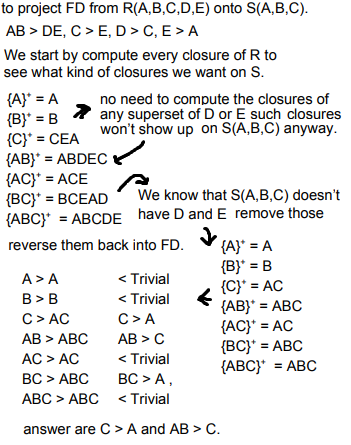
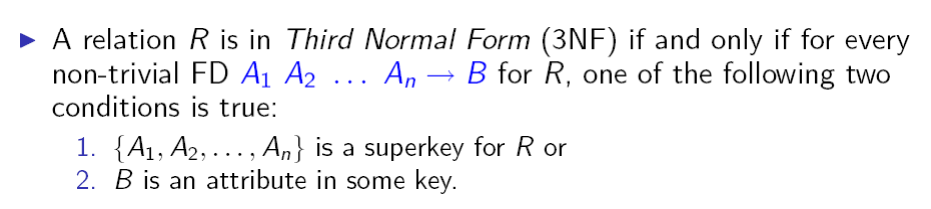
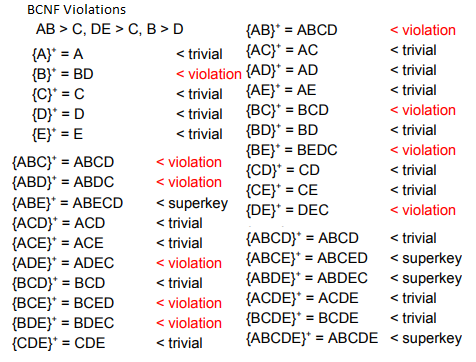
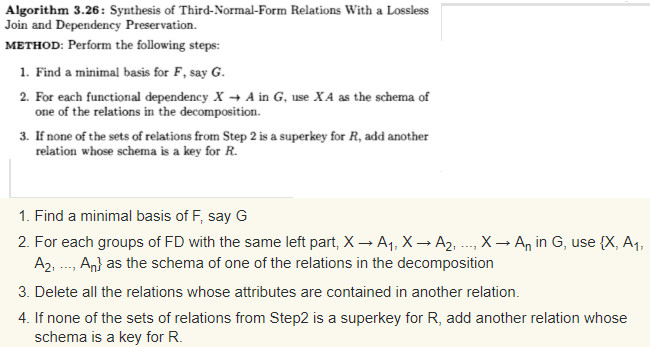
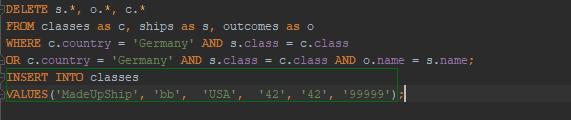
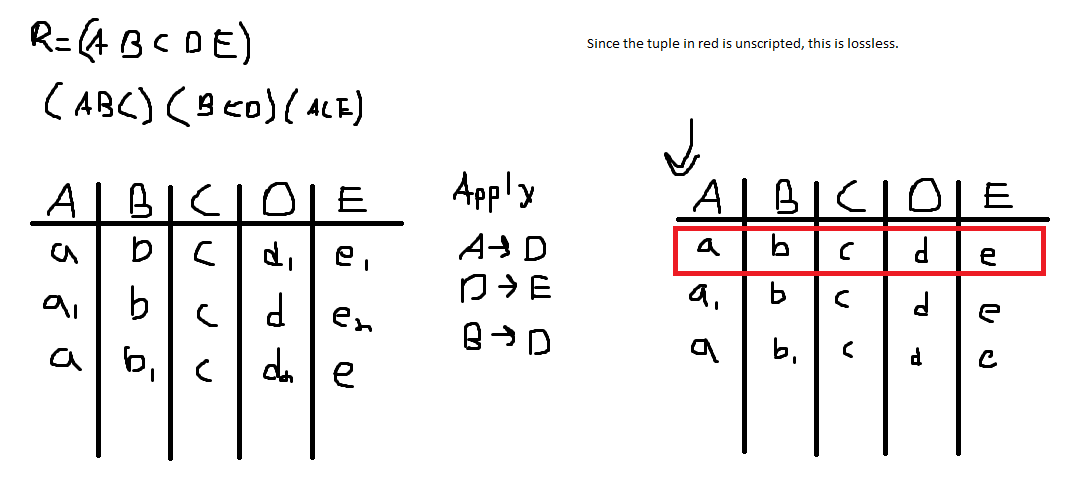
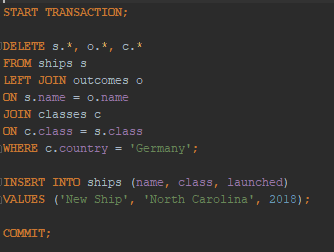
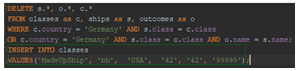
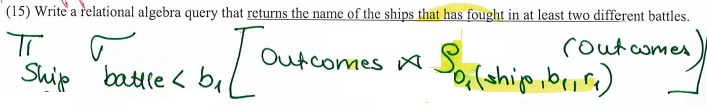
**Database Management Systems** - Collection of interrelated data and set of programs to access the data  
**Why are DBMS widely used?** Efficiency, ease of developing DB,   
**Describe physical / logical / view level abstraction**: **View level**: application programs hide details of data types, **Logical level**: What is the data? Data types and such **Physical level**: How is the data stored?  
**Data Model** - Collection of tools for describing data, relationships of data items, semantics. Can be shown as ER models  
**Data Definition Language (DDL)** - Defines the database schema and constraints  
**Data Manipulation Language (DML)** - Language to access and manipulate the data. A query language.  
**Procedural Query Language**: user specifies what data is required and how to get that data  
**Non procedural Query Language**: user specifies what data is required without how to get  
**Transaction** - Unit of work to be executed atomically and in isolation  
**Serializability** – An ability to perform multiple transactions at the same time without affecting each transaction result  
**2 phase locking** – method to guarantee serializability by locking data if any transaction modify data which can affect results of other transactions  
**Concurrency control is important to ensure consistency**  
***A*tomicity**: all-or-nothing of the transaction’s effect will take place  
***C*onsistency**: each transaction leaves the system in a consistent state  
***I*solation**: each transaction must appear to be executed as if no others are executed at the same time  
***D*urabilit**y: effect of a transaction must never be lost after the transaction is completed  
**Relation** – a table with columns and tuples **Database** – collection of relations  
**Relation schema**: relation name and attribute list  
**database schema** - set of all relation schemas in the database  
**SQL** is a query language with a data definition component; it is both a **DML** and a **DDL**, and it is **non-procedural language**  
A **key** is a single (or list of) attributes that can differentiate two nearly identical tuples. If a relation has a key, then no two tuples may match in all attributes. There can only be one **PRIMARY KEY** per relation, but multiple **UNIQUE** keys are allowed.  
**Super key:** key with extra unneeded attributes **Candidate Key:** minimal key **Primary key:** candidate key chosen to be a key  
**Create a Relation**:  **Delete a Relation**: 

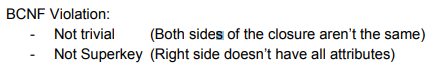
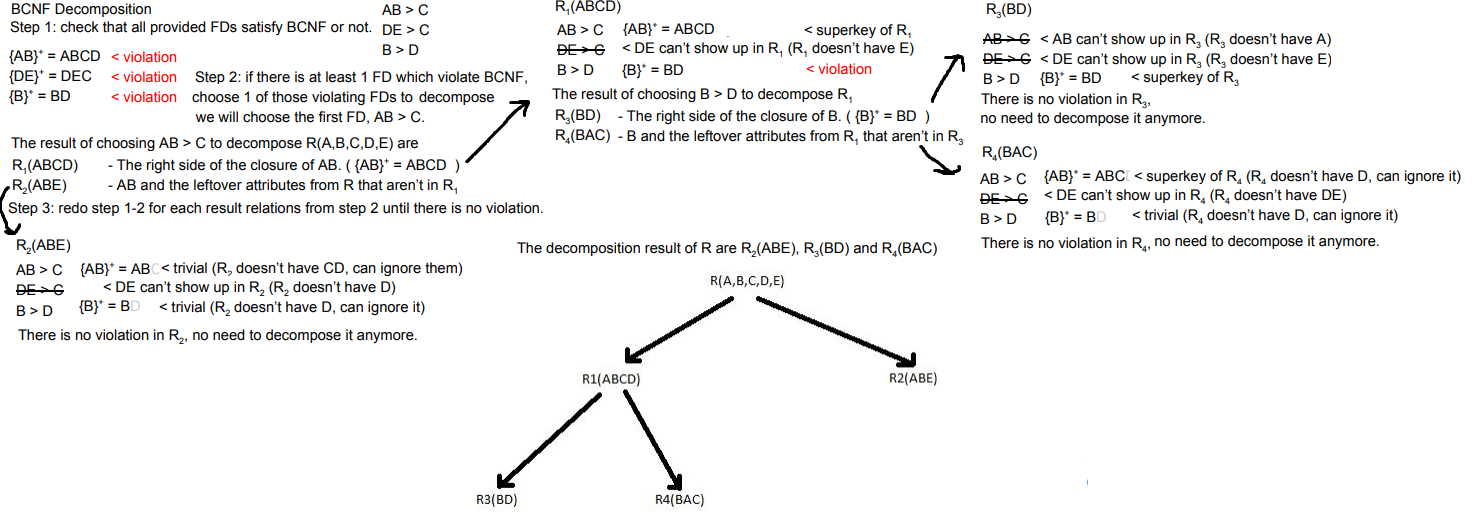
**Attribute** – basic element in a relation (column). **Data type**s include: INT, FLOAT, CHAR(n) VARCHAR(n), DATE, TIME  
**Relational Algebra** – Mathematical system including operands and operators  
**Operands** – variables or values **Operators** – symbols denoting procedures  
**Union, Intersection and Difference** work exactly as you would think, but both operands MUST have the same relation schema  
R1 := **σ*C***(R2**) selection** ( all the tuples of R2 that satisfy condition c )  
R1 := **π*L***(R2**) projection** ( all the tuples of R2 with only the attributes on the list L, removing duplicates )  
R3 := R1 **Χ** R2 **cartesian product** ( as per usual, but it is R1attributes+R2attributes in that order)  
R3 := R1 **⋈*C*** **R2 theta join** (cartesian product then selection)  
R3 := R1 **⋈** R2 **natural join** ( theta join but w/o selection. Will equate attributes of same name as well)  
R1 := **ρ**R1(A1,…,A*n*)(R2) **rename** ( use to rename an attribute )  
**Bag** – a set, but an element can appear more than once **SQL Query** consists of: **SELECT FROM WHERE**  
**Rename in SQL** : SELECT ‘name’ AS ‘newname’ **Conditionals Allowed in the WHERE clause: AND OR NOT = <> < > <= >= ** 

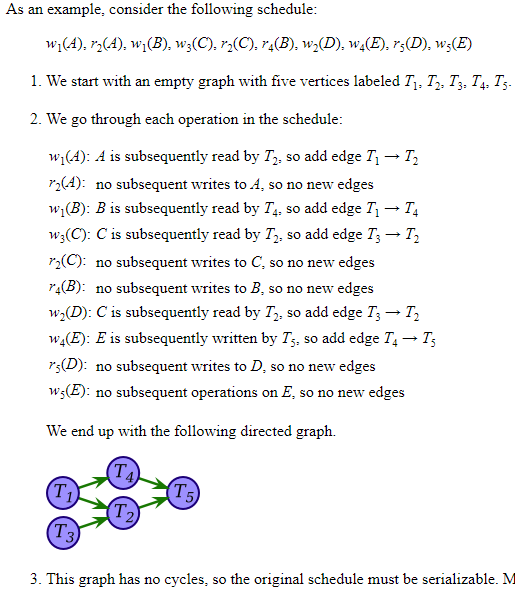
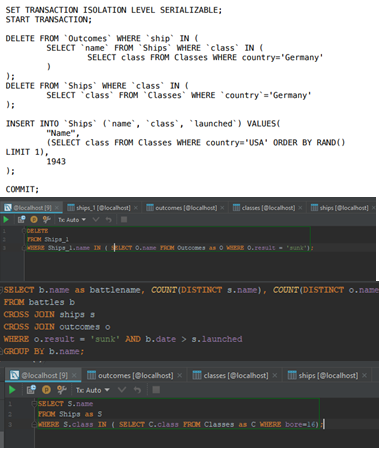


  
**Constraints**: keys, not null, unique, check. **Referential integrity constraints**: value appearing in 1 context must also appear in another related context  
**CREATE VIEW (security)** for the Students (name, email, major GPA) (how to make GPA private only?)  
**CREATE** **VIEW** Student \_PUBLIC **AS** **SELECT** name, email, major **FROM** Student; **Problem with Views?** If a virtual view is updated, the updates must be propagated back to the base relations, which can become difficult.   
**Normalization (FD, Projecting FD, Boyce-Codd Normal Form(BCNF), Third Normal Form)** 

 ****  
********



****